# Esercitazione di Fine Settimana – Week 2

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Gian Mario |
|  |  | Cognome | Falchi |
|  |  | Data | 17/09/2021 |

Leggete attentamente ogni domanda e argomentare quanto più possibile fornendo anche degli esempi.

1. Descrivere le modalità per ritornare più valori da un metodo in C#

I modi per restituire valori sono:

* restituire un dato strutturato tramite return (array, lista, struct, tupla)
* restituire valori tramite out
* combinazione delle 2 precedenti

Es. static int[] Prova() //restituisco un vettore di int

{

int[] a = { 1, 2, 3, 4 };

return a;

}

static List<int > Prova2() //uguale ma con le liste

{

List<int> a = new List<int> { 5, 6, 8, 7 };

return a;

}

static ProvaOgg Prova3() //restituisco un tipo di dato

//strutturato

{ //(struct o oggetto)

Prova a=new(); //restituisco i valori tramite i campi

a.p1 = 11;

a.p2 = 12;

a.p3 = 13;

return a;

}

static (int val1, int val2) Prova4()

{ //tramite la sintassi delle tuple

return (21, 22); //posso restituire più valori assieme

}

static void Prova4 (int a, out int doppio, out int triplo)

{

doppio = 2 \* a; //restituisco valori tramite out

triplo = 3 \* a;

}

Ovviamente posso combinare return e out, per esempio

static bool Prova4 (int a, out int doppio, out int triplo)

{

doppio = 2 \* a;

triplo = 3 \* a;

return false;

}

Sintassi simile ha il metodo TryParse.

1. Descrivere le due tipologie di casting tra tipi in C#

Il casting implicito da un tipo A a un tipo B non ha bisogno di sintassi

Es da int a double

Però se I due tipi non sono perfettamente compatibili potrei avere perdita di dati, come nel caso da double a int

Il casting esplicito risolve quest’ultimo problema, e può essere già definito nelle librerie (come il casting tra i tipi base) oppure possiamo definirlo noi per tipi creati da noi

public static implicit operator ComplexNumber(double a)

{

ComplexNumber b = new(a, 0);

return b

}

public static explicit operator double(ComplexNumber a)

{

return a.Modulo;

}

Il casting implicito può essere effettuato senza sintassi, quindi

double a = 4.3;

ComplexNumber b= new();

b=a;

Il casting esplicito ha la sintassi seguente

ComplexNumber a = new (10, -7);

double b= (double)a;

Entrambi verranno convertiti secondo la logica che abbiamo scelto noi

1. Quali sono gli utilizzi della keyword static?

La parola chiave static serve per dichiarare un membro statico di una classe.

Un’istanza di una classe contiene la “copia” di tutti i membri della classe che non sono statici, mentre dei membri statici ne esiste solo una copia.

I membri non statici possono essere usati tramite la sintassi Istanza.Membro

Mentre i membri non statici tramite la sintassi Tipo.Membro

Un caso classico è quello di Console.Writeline

Dove Writeline è un metodo statico della classe console

1. Descrivere le modalità di implementazione di un evento in C#

Se l’avessi capito non avrei toppato nell’esercitazione del 16

1. A cosa serve l’interfaccia IEnumerable<T>? Come si implementa in una nostra classe?

IEnumerable<T> è l’interfaccia di base per le collezioni generiche, per esempio List<T>, e ci permette di effettuare iterazioni su esse tramite il costrutto foreach.

Ci fornisce i due metodi:

public IEnumerator<T> GetEnumerator()

IEnumerator IEnumerable.GetEnumerator()

Implementandoli in questo modo

public IEnumerator<T> GetEnumerator()

{

return GetEnumerator();

}

IEnumerator IEnumerable.GetEnumerator()

{

foreach (var item in items)

{

yield return item;

}

}

…possiamo utilizzare il foreach su una lista incapsulata nella stessa classe:

private List<T> items = new();

**Esercitazione Pratica**

* Realizzare una classe Warehouse per gestire un Magazzino Merci, con le seguenti proprietà:
  + *Id Magazzino (GUID)*
  + *Indirizzo*
  + *Importo Totale Merci in giacenza*
  + *Data Ultima Operazione*
  + *Lista delle Merci in giacenza*
* Realizzare **l’overload degli operatori + e –** in modo che sia possibile aggiungere e rimuovere Merci dalla lista (l’overload dovrà anche occuparsi di aggiornare l’Importo Totale e la Data di Ultima Operazione)
* Realizzare un metodo StockList() che stampi i dati del Magazzino, inclusa la lista delle Merci in giacenza
* Realizzare una gerarchia di classi per rappresentare le Merci (Good). Tutte le classi avranno le proprietà
  + *Codice Merce*
  + *Descrizione*
  + *Prezzo*
  + *Data di Ricevimento*
  + *Quantità in Giacenza*
  + Realizzare le classi che rappresentano:
    - ElectronicGood, con la proprietà aggiuntiva *Produttore*
    - PerishableGood, con le proprietà aggiuntive *Data di Scadenza* e *Modalità di Conservazione* (enum con i valori FREEZER, FRIDGE e SHELF)
    - SpiritDrinkGood, con le proprietà aggiuntive *Tipo* (enum con i valori WHISKY, WODKA, GRAPPA, GIN e OTHER) e *Gradazione Alcoolica*
  + Tutte le classi saranno dotate di costruttore che accetti tutti i parametri necessari per popolare le proprietà
  + Tutte le classi dovranno implementare la propria versione del metodo ToString() e visualizzare tutti i dati //fatto fino a qua
  + ***OPZIONALE***: realizzare una procedura di caricamento dati da un file della lista delle Merci in giacenza (l’implementazione di eventi per notificare le fasi di caricamento dati costituisce un bonus)
  + ***OPZIONALE 2***: utilizzare una (o più) eccezione custom per gestire tutte le tipologie di errori che dovessero verificarsi durante l’utilizzo della classe Warehouse e delle classi Good
* Realizzare una Console app che
  + Crei un nuovo Magazzino
  + Permetta di ricevere diverse tipologie di Merci (gestire l’input dall'utente)
  + Stampi i dati del Magazzino e le Merci in giacenza